(19) Japan Patent Office (JP)

(12) Publication of Unexamined Patent Application (A)

(11) Japanese Patent Application Laid-Open Publication Number: Tokukai-Hei 11-73516

(43) Laid-Open Date: Heisei 11-3-16 (March 16, 1999)

(51) Int.Cl.[6]		Identification Code	FI		
G06T	11/00		G06F	15/72	G
	1/00			15/62	A

Request for Examination: Not requested

Number of Claims: 7

OL (24 pages in total)

(21) Application Number: Tokugan-Hei 9-231541

(22) Filing Date: Heisei 9-8-27 (August 27, 1997)

(71) Applicant: 000005496

Fuji Xerox Co., Ltd.

2-17-22, Akasaka, Minato-ku

Tokyo, Japan

(72) Inventor: Hiroshi Okubo

c/o Fuji Xerox Co., Ltd. Ebina Center

2274, Hongou, Ebina-shi,

Kanagawa Prefecture

(74) Representative: Patent Attorney; Jun Nakajima (and four others)

(54) [Title of the Invention] IMAGE PROCESSING APPARATUS

[Abstract]

[Object] To improve a processing speed for converting intermediate code image data into raster data at the time of an output while reducing an amount of memory usage for the intermediate code image data.

[Solving Means] At the time of printing out, intermediate code image data are reconstructed by scanning, along a predetermined scanning direction, edge lists which are created and stored so as to eliminate overlapping portions from the intermediate code image data. At this time, when the type of object of the intermediate code image data is color graphics, for example, a possibility that the shape of the object becomes rectangular is high even after removing an overlapped portion with another object. Thus, by scanning the edge lists in the sub scanning line direction as the scanning direction as shown in Fig. 16(C), intermediate code image data of a small amount of data such as rectangular display lists, for example, can be reconstructed. The amount of data of the intermediate code image data to be reconstructed can be reduced since the scanning direction is switched to an appropriate scanning direction in accordance with the type of object as in the manner described above.

[Scope of Claims]

[Claim 1] An image processing apparatus comprising:

conversion means which converts page description data into intermediate code image data, the page description data representing each of a plurality of objects to be rendered in layers in accordance with a predetermined rendering sequence in a predetermined rendering area;

storage means which stores the converted intermediate code image data on each of the objects in a memory;

updating means which creates edge lists for each of the objects in each of predetermined scanning lines corresponding to the intermediate code image data by retracing the rendering sequence of the object so that image areas represented by the intermediate code image data on the respective objects would not overlap with one another, and which updates the previously-stored intermediate code image data on each of the objects by using the edge lists created for each of the objects; and

reconstruction means which reconstructs the intermediate code image data on each of the objects from the edge lists, by scanning the edge lists for each of the objects while switching scanning directions from one to another in accordance with a predetermined condition.

[Claim 2] The image processing apparatus according to claim 1, characterized in that the predetermined condition is a type of object.

[Claim 3] The image processing apparatus according to any one of claims 1 and 2, characterized in that, in accordance with the predetermined condition, the reconstruction means switches scanning to any one of a mode in which the scanning direction is set to a main scanning line direction, a mode in which the scanning direction is set to a sub scanning line direction, and a mode in which the scanning direction is switched between the main

scanning line direction and the sub scanning line direction for each of the edge lists in accordance with a predetermined switching condition.

[Claim 4] The image processing apparatus according to claim 3, characterized in that the predetermined switching condition is a distance between an edge list of a scanning target and an edge list to become the next scanning target.

[Claim 5] An image processing apparatus comprising:

conversion means which converts page description data into intermediate code image data, the page description data representing each of a plurality of objects to be rendered in layers in accordance with a predetermined rendering sequence in a predetermined rendering area;

storage means which stores the converted intermediate code image data for each of the objects in a memory;

graphics processing means which creates edge lists for each of the objects in each of predetermined scanning lines corresponding to the intermediate image data by retracing a rendering sequence of the objects so that image areas represented by the intermediate code image data on the respective objects would not overlap with one another, and which executes overlapped graphics processing of updating the stored intermediate code image data on each of the objects by using the edge lists created for each of the objects; and

control means which controls the overlapped graphics processing so as not to be performed in a case where the type of object is a specific type.

[Claim 6] An image processing apparatus comprising:

conversion means which converts page description data representing each of a plurality of objects to be rendered in layers in accordance with a predetermined rendering sequence in a predetermined rendering area into

intermediate code image data;

storage means which stores the converted intermediate code image data for each of the objects in a memory;

graphics processing means which creates edge lists for each of predetermined scanning lines corresponding to the intermediate image data for each of the objects by retracing a rendering sequence of the objects so that overlapped portions of image areas represented by the intermediate code image data for each of the object are eliminated, and which executes overlapped graphics processing updating the stored intermediate code image data for each of the objects by the edge lists created for each of the objects; and

control means which controls the overlapped graphics processing so as not to be performed on the plurality of objects in a case where an overlapping state between a plurality of objects is equal to or greater than a predetermined level.

[Claim 7] The image processing apparatus according to any one of claims 5 and 6, characterized in that, in a case where the plurality of objects exist for which the overlapped graphic processing is prohibited, the control means controls

the overlapped graphics processing so that the graphics processing means executes the overlapped graphic processing on an overlapped portion only between the objects for which the overlapped graphics processing is prohibited .

[Detailed Description of the Invention]

[0001]

[Technical Field to which the Invention Pertains] The present invention relates to an image processing apparatus. To be more specific, the present invention relates to an image processing apparatus including a function to create edge lists from intermediate code image data which represent each of multiple objects to be rendered in layers in accordance with a predetermined rendering sequence, and a function to reconstruct the intermediate code image data from the edge lists.

[0002] It should be noted that the aforementioned objects mean objects to be rendered such as graphics, fonts and images. Moreover, page description data mean image data related to each of the objects represented in page description language.

[0003]

[Prior Art] Conventionally, in image processing to create raster data on the basis of page description data which represent objects to be rendered, there is proposed a method of creating raster data on the basis of the intermediate code image data obtained after the page description data of objects are temporarily converted into the intermediate code image data. Intermediate code image data have an advantage of being capable of representing objects with a small amount of data. Thus, an amount of memory to be used in the image processing can be saved by the aforementioned method.

[0004] As an example of using such intermediate code image data, in Japanese Unexamined Patent Application Publication No. Hei 6-284297, disclosed is the technique for creating display lists of each of objects by always checking whether or not each object overlaps with another object whose display list has already been created when intermediate code image data (display lists in a case of this patent

document) of each object are created. Precisely, in a case where the object in the intermediate code image data overlaps with a different type of object, the display lists of the object are reconstructed so that the object would not overlap with the different kind of object (it is to be noted that the objects herein are assumed to be held in an compressed format, not in an intermediate code image data format). According to this technique, a different compression can be used for each of the objects, and further, the memory write time for raster data is reduced since the overlapping of display lists is eliminated.

[0005] However, in the aforementioned technique, there is a drawback that a considerable amount of time is consumed for checking on the overlapping of each of objects and for reconstructing display lists when intermediate code image data for each of the objects are created, in a case where there are a large number of objects per page.

[0006] Furthermore, in Japanese Unexamined Patent Application Publication No. Hei 7-170411, disclosed is the technique for causing the image processing apparatus, which is configured to convert image data into raster image and then output the raster image, to encode image data having continuity by use of the function representing continuity.

[0007] This technique makes it possible to reduce the amount of memory usage for intermediate code image data and to reduce the processing time to create the intermediate code image data as follows. Specifically, this technique is applied to edge lists each having an identical run length, and placed at equal intervals in a sub scanning line direction when the edge lists are to be scanned in the sub scanning line direction. This scanning is performed in order to reconstruct the intermediate code image data from an object targeted for processing after the execution of the processing in which

the intermediate code image data are processed so that objects would not overlap with one another (hereinafter, this processing is referred to as overlapped graphics processing). However, there is no effect in the reduction in the amount of memory

usage or in the processing time to create intermediate code image data on edge lists other than ones each having an identical run length and placed at equally intervals in the sub scanning line direction. In this case, the processing time becomes even longer since the determination process is executed to find out whether or not the edge lists are placed at equal intervals in the sub scanning line and the run lengths thereof are identical.

[0008] Moreover, the present applicant has filed the patent application for the invention relating to the image processing method which eliminates overlapping of each object in Japanese Patent Application No. Hei 8-276652. In this method, intermediate code image data on each object converted from page description data is stored, and then outline data representing an outline of an object on an upper layer which masks an object to be rendered on the lower layer are created for each object by retracing from the uppermost layer. Subsequently, the intermediate code image data are reconstructed from portions remained without being masked.

[0009] By use of this technique, the overlapped graphics processing does not have to be started every time when intermediate code image data are created unlike in a system of the technique described in aforementioned Japanese Unexamined Patent Application Publication No. Hei 6-284297. The overlapped graphics processing is started only in a case where there is a lack of storage space for intermediate code image data, and where the writing of raster data on memory is behind. Thus, by performing the processing collectively or the processing on a predetermined part, the processing speed can be improved.

[0010] However, in the technique proposed in Japanese Patent Application Laid-open Publication No. Hei 8-276652, the overlapped graphics processing is performed without recognizing the contents of objects. For this reason, in a case where intermediate code image data are created from respective edge lists of objects which have been subjected to the overlapped graphics processing, sizes of the created

intermediate code image data depend on the types of objects, and thereby become different from one another. Moreover, in a case where a number of objects overlap with a single object, an object on a lower layer may be segmented, and then the amount of data of the created intermediate code image data may increase. Thereby, there is a concern that the increase in the amount of data causes problems of an increase in the amount of memory usage and a reduction in the processing speed for expansion to raster data.

[0011]

[Problems to be Solved by the Invention] The present invention has been made to solve the aforementioned problems. A first object of the present invention is to provide an image processing apparatus capable of reducing an amount of memory usage of intermediate code image data, and of improving an expansion processing speed to raster data during an output. A second object of the present invention is to provide an image processing apparatus capable of preventing an increase in the amount of memory usage of intermediate code image data and a reduction in the expansion processing speed caused by segmentalization of objects due to overlapped graphics processing.

[0012]

[Means for Solving the Problems] In order to achieve the first object, an image processing apparatus as recited in claim 1 is characterized as follows. The image processing apparatus includes conversion means which converts page description data representing each of multiple objects to be rendered in layers in accordance with a predetermined rendering sequence in a predetermined rendering area into intermediate code image data; storage means which stores the converted intermediate code image data for each of the objects in a memory; updating means which creates edge lists for each

of predetermined scanning lines corresponding to the intermediate code image data for each of the objects by retracing a rendering sequence of the objects so that overlapped portions of image areas represented by the intermediate code image data for each of the objects are eliminated, and which updates the stored intermediate code image data for each of the objects by the edge lists created for each of the objects; and reconstruction means which reconstructs, by scanning the edge lists for each of the objects updated by the updating means while switching scanning directions from one to another in accordance with a predetermined condition, intermediate code image data for each of the objects from the edge lists.

[0013] The image processing apparatus as recited in claim 2, is characterized in that the predetermined condition is a type of object in the image processing apparatus as recited in claim 1.

[0014] Furthermore, the image processing apparatus as recited in claim 3 is characterized in that, in accordance with the predetermined condition, the reconstruction means switches scanning to any one of a mode in which the scanning direction is set to a main scanning line direction, a mode in which the scanning direction is set to a sub scanning line direction, and a mode in which the scanning direction is switched between the main scanning line direction and the sub scanning line direction for each of the edge lists in accordance with a predetermined switching condition in the image processing apparatus as recited in any one of claims 1 and 2.

[0015] The image processing apparatus as recited in claim 4 is characterized in that the predetermined switching direction is a distance between an edge list of a scanning target and an edge list to become the next scanning target in the image processing apparatus as recited in claim 3.

[0016] In order to achieve the second object, an image processing apparatus as recited in claim 5 is characterized as follows. The image processing apparatus includes: conversion means which converts page description data into intermediate code image data, the page description data representing each of a plurality of objects to be rendered in layers in accordance with a predetermined rendering sequence in a predetermined rendering area; storage means which stores the converted intermediate code image data on each of the objects in a memory; graphics processing means which creates edge lists for each of the objects in each of predetermined scanning lines corresponding to the intermediate image data by retracing a rendering sequence of the objects so that image areas represented by the intermediate code image data on the respective objects would not overlap with one another, and which executes overlapped graphics processing of updating the stored intermediate code image data on each of the objects by using the edge lists created for each of the objects; and control means which controls the overlapped graphics processing so as not to be performed in a case where the type of object is a specific type.

[0017] Moreover, in order to achieve the second object, an image processing apparatus as recited in 6 is characterized as follows. The image processing apparatus includes: conversion means which converts page description data representing each of a plurality of objects to be rendered in layers in accordance with a predetermined rendering sequence in a predetermined rendering area into intermediate code image data; storage means which stores the converted intermediate code image data on each of the objects in a memory; graphics processing means which creates edge lists for each of predetermined scanning lines corresponding to the intermediate image data on each of the objects by retracing a rendering

sequence of the objects so that overlapped portions of image areas represented by the intermediate code image data on each of the object are eliminated, and which executes overlapped graphics processing updating the stored intermediate code image data on each of the objects by the edge lists created for each of the objects; and control means which controls the overlapped graphics processing so as not to be performed on the plurality of objects in a case where an overlapping state between a plurality of objects is equal to or greater than a predetermined level.

[0018] Moreover, the image processing apparatus as recited in claim 7 is characterized in that, in a case where the plurality of objects exist for which the overlapped graphic processing is prohibited, the control means controls the overlapped graphics processing so that the graphics processing means executes the overlapped graphic processing on an overlapped portion only between the objects for which the overlapped graphics processing is prohibited as recited in any one of claims 5 and 6.

[0019] In the image processing apparatus as recited in claim 1, page description data representing each of a plurality of objects to be rendered in layers in accordance with a predetermined rendering sequence in a predetermined rendering area are converted into intermediate code image data by the conversion means. Then, the converted intermediate code image data on each of the objects are stored in a memory by the storage means. It should be noted that as the intermediate code image data, edge lists, display lists, run length lists, or the like to be described later may be used.

[0020] Then, the updating means creates edge lists for predetermined scanning lines corresponding to the intermediate image data on each of the objects by retracing a rendering sequence of the objects so that overlapped

portions of image areas represented by the intermediate code image data on each of the objects are eliminated, and then updates the stored intermediate code image data on each of the objects by the edge lists created for each of the objects.

[0021] Here, the updating means, for example, by retracing the rendering sequence of the objects, creates outline data representing at least an outline of an object on the upper layer side, which masks an object to be rendered on the lower layer side. By removing the created outline data area from the intermediate code image data on each of the objects, the updating means can create edge lists on each of the objects by retracing the rendering sequence of the objects, so that the overlapped portions of the aforementioned image area are eliminated. Then, the updating means updates the stored intermediate code image data on each of the objects by the edge lists for each of the objects.

[0022] In this example, an amount of data of the intermediate code image data after the update is reduced as compared to that of the intermediate code image data before the update in many cases. To be specific, the intermediate code image data after the update is one in which the outline data area of an object on a layer higher than each of the objects are removed from the intermediate code image data on each of the objects. Thus, the amount of data of the intermediate code image data stored in the memory is reduced by the aforementioned image processing method.

[0023] Furthermore, in the present invention, intermediate code image data are reconstructed for each of the objects from the edge lists with the reconstruction means by scanning the updated edge lists for each of the objects, while switching scanning directions from one to another in accordance with a predetermined condition.

[0024] To be more specific, as recited in claims 2 and 3, the updated edge lists on each of the objects are scanned by the reconstruction means while switching, in accordance with the type of object, any one of a mode in which the scanning direction is set to the main scanning line direction, a mode in which the scanning direction is set to the sub scanning line direction, and a mode in which the scanning direction is switched to one of the main scanning line direction and the sub scanning line direction for each of the edge lists in accordance with a predetermined switching direction. Thereby, the intermediate code image data are reconstructed on each of the objects from the edge lists.

[0025] For example, in a case where the type of object is font or black and white graphics, since there is a possibility that the shape of the object after overlapping portions are removed becomes quite complicated, the edge lists are preferably scanned while switching to the mode in which the scanning direction is switched to one of the main scanning line direction and the sub scanning line direction for each of the edge lists in accordance with the predetermined switching condition (for example, the distance between the edge list of the scanning target and the edge list to become the next scanning target as recited in claim 4). As one example, in a case where the distance to the edge list to become the next scanning target along the sub scanning line direction is shorter than the distance to the edge list to become the next scanning target along the main scanning line direction when viewed from the edge list of the scanning target, an amount of data of distance information between the edge list and an adjacent edge list can be small by scanning the data while switching the scanning direction to the sub scanning line direction. Thus, the memory size for storing the data can be reduced, and the memory space can be saved.

[0026] Moreover, in a case where the type of object is color graphics, differently from the case of a black and white graphics, it is highly likely that the shape of the object becomes rectangular even after the overlapped portions with another object are removed. Thus, by scanning the edge lists while switching to the mode in which the scanning direction is set to the sub scanning line direction, it is possible to reconstruct the edge lists as the intermediate code image data having a small amount of data such as rectangular display lists or codes using functions to represent continuity, for example. Thereby, the memory size for storing the data can be reduced, and the memory space can be saved.

[0027] It should be noted that in a case where the type of object is image or pattern image (one constituted of a plurality of images patterned in a predetermined rule; hereinafter, an image or a pattern image is correctively called as an image), the intermediate code image data merely include clip information, and the entity of the image is retained in a state of being as it is or being compressed in another memory area. For this reason, in the expansion process, the image needs to be read in the sequence of addresses from the area where the entity of the image is retained, and then to be clipped with the intermediate code image data (= set of edge lists) sorted in the main scanning line direction and the sub scanning line direction, and then, to be written into the memory. Thus, it is preferable to scan the edge lists while switching to the mode in which the scanning direction is set to the main scanning line direction.

[0028] As has been described so far, in accordance with the type of object, updated edge lists on each of the objects are scanned while a scanning mode is switched to any one of the mode in which the scanning direction is set to the main scanning line direction, the mode in which the scanning direction

is set to the sub scanning line direction, and a mode in which the scanning direction is switched to one of the main scanning line direction and the sub scanning line direction on each of the edges list in accordance with the predetermined switching condition. Thus, the intermediate code image data on each of the objects are reconstructed from the edge lists. Thereby, the amount of data of the intermediate code image data can be reduced. Thus, the amount of memory usage can be reduced while the expansion processing speed to raster data at the time of an output can be improved.

[0029] Next, in the image processing apparatus as recited in claim 5, page description data representing each of a plurality of objects to be rendered in layers in accordance with a predetermined rendering sequence in a predetermined rendering area are converted into intermediate code image data by the conversion means. Then, the converted intermediate code image data on each of the objects are stored in a memory by the storage means.

[0030] Then, the graphics processing means creates edge lists for each of predetermined scanning lines corresponding to the intermediate code image data on each of the objects by retracing a rendering sequence of the objects so that overlapped portions of image areas represented by the intermediate code image data on each of the objects are eliminated. The graphics processing means then executes overlapped graphics processing which updates the stored intermediate code image data for each of the objects by the edge lists created for each of the objects. Specifically, the overlapped graphics processing is constituted of two processes including: the creation of edge lists for each of the objects so that the overlapped portions of image areas represented by the intermediate code image data on each of the object are eliminated; and the updating of the intermediate code image data by the

created edge lists.

[0031] However, in a case where the overlapped graphics processing is to be executed on a certain object, and in a case where the type of the certain object (that is, the object to become the base) is a specific type, the control means controls to prohibit the overlapped graphics processing to be performed on the certain object.

[0032] For example, in a case where the type of object is image, the intermediate code image data thereof merely include clip information, as described above, and the entity of the image is retained in another memory in the state of being as it is or of being compressed. For this reason, in the expansion process, the image needs to be read in the sequence of the addresses from the area where the entity of the image is retained, and to be clipped with the intermediate code image data (= set of the edge lists) sorted in the main scanning line direction and the sub scanning line direction, and then, to be written into the memory.

[0033] Thus, in the case where the type of object is image, the overlapped graphics processing by the graphics processing means is preferably prohibited by the control means.

[0034] As has been described above, by prohibiting the overlapped graphics processing on an object for which it is not appropriate to execute the overlapped graphics processing, it is possible to prevent an increase in the amount of memory usage of the intermediate code image data and reduction in the expansion processing speed due to the segmentalization of the object.

[0035] Furthermore, in the image processing apparatus as recited in claim 6, as in the image processing apparatus as recited in the aforementioned claim 5, the graphics processing means creates the edge lists for each of

predetermined scanning lines corresponding to intermediate image data for each of the objects by retracing a rendering sequence of the objects so that overlapped portions of image areas represented by the intermediate code image data for each of the objects are eliminated. Then, the overlapped graphics processing which updates the stored intermediate code image data for each of the objects by the edge lists created for each of the objects is executed by the graphics processing means.

[0036] However, in a case where the degree of the overlapped states between the plurality of objects is equal to or greater than a predetermined level, the control means controls to prohibit the overlapped graphics processing from being performed on the plurality of objects. For example, in a case where the number of times that edge lists representing an object are cut by another object is equal to or greater than a predetermined value, it is considered that the segmentalization of the object has occurred. Thus, it is desirable to prohibit the overlapped graphics processing from being executed for the object.

[0037] As described above, by prohibiting the overlapped graphics processing from being performed on an object in which the degree of the overlapped states with another object is equal to or greater than a predetermined level, it is possible to prevent an increase in the amount of memory usage of the intermediate code image data and a reduction in the expansion processing speed from occurring due to the segmentalization of the object in advance.

[0038] However, in the overlapped graphics processing, the edge lists created by retracing the rendering sequence of the object so that the overlapped portions of the image area are eliminated are, and then the intermediate code image data are updated by the edge lists. For this

reason, in a case where there exist a plurality of objects for which the overlapped graphics processing is prohibited, an object to be processed later needs to be rendered in advance. Thus, there is a case where the intermediate code image data representing each of the objects needs to be temporarily stored in a working area of the memory, and thereafter the rendering sequence needs to be changed, and then, the temporarily stored intermediate code image data need to be stored again in an official memory.

[0039] Thus, as recited in claim 7, in the case where there exist a plurality of objects for which the overlapped graphics processing is prohibited, it is desirable to control, by the control means, the overlapped graphics processing to be executed only among the objects for which the overlapped graphics processing is prohibited. Thereby, it is no longer necessary that the intermediate code image data are temporarily stored in a working area of the memory, thereafter that the rendering sequence is changed, and then that the temporarily stored intermediate code image data are stored again in an official memory, as described above. Thus, the processing efficiency can be improved.

[Means for Solving the Problems]

[0040]

[Embodiment Modes for Carrying Out the Invention] Hereinafter, various embodiments according to the present invention will be described in detail while referring to drawings.

[0041] [First Embodiment] To begin with, a first embodiment corresponding to the invention of claims 1 to 4 as recited in scope of claims will be explained.

[0042] (Configuration of Image Processing Apparatus) The configuration of an image processing apparatus in the present embodiment

will be described below. As shown in Fig. 1, an image processing apparatus 10 is provided with an upper level processor 12, an overlapped graphics processor 14, a list storage unit 11, an expansion unit 16, a schedule manger 18, a measurement unit 20, band buffers 26, a printer engine 30, a transfer unit 28, a band buffer manager 22, and an error processor 24. The upper level processor 12 interprets code image data described in page description language by a decomposer embedded therein. The upper level processor 12 creates intermediate code image data being divided in band units in a case where the code image data are objects to be rendered such as graphics, fonts, or images (hereinafter, termed as objects). The overlapped graphics processor 14 performs the overlapped graphics processing to eliminate overlapping between objects in units of band. The list storage unit 11 serves as a memory for storing edge lists which are created in the overlapped graphics processing. The expansion unit 16 expands intermediate code image data in each of the bands to raster data. The schedule manager 18 determines, based on information (such as an address of intermediate code image data retaining area within the memory area) related to the intermediate code image data in each of the bands, whether or not the intermediate code image data can be processed to be expanded in real-time. The measurement unit 20 measures or estimates a period of time required for expanding the intermediate code image data in each of the bands. The band buffers 26 store therein the raster data in each of the bands, which are created in the expansion process by the expansion unit 16. The printer engine 30 prints out the raster data. The transfer unit 28 transfers the raster data stored in the band buffers 26 to the printer engine 30. The band buffer manager 22 manages use conditions of the band buffers 26. The error processor 24 processes errors when an

error occurs in the expansion process of raster data or the like.

[0043] The image processing apparatus 10 configured in this manner is started in a case where code image data is input thereto from an external image processing apparatus via a network, or from a spool disk. Then, in the image processing apparatus 10, the code image data input thereto are sequentially interpreted by the decomposer within the upper level processor 12. As described above, in a case where the code image data are objects, intermediate code image data divided in units of band are created by the upper level processor 12. The created intermediate code image data in units of band are retained respectively by bands in the intermediate code image data storage area in a memory area embedded in the upper level processor 12. Incidentally, display lists, run length lists, or the like can be cited as an example of the intermediate code image data, here.

[0044] During the aforementioned processing, in a case where there is a lack of memory in the intermediate code image data storage area, the overlapped graphics processor 14 is started by the upper level processor 12. Then, the overlapped graphics processing (to be described later in detail) which eliminates overlapping between image areas represented by the intermediate code image data in units of band is started. This overlapped graphics processing is executed until the lack of memory in the intermediate code image data storage area is resolved.

[0045] The upper level processor 12 notifies the expansion unit 16 of information (such as addresses of the intermediate code image data storage area within a memory area) related to the intermediate code image data for each of the bands after the processing of an amount of page description language for one page is completed. Then, the expansion unit 16 causes the schedule manager 18 to determine whether or not the intermediate code

image data for each of the bands can be expanded to raster data in real-time.

[0046] This schedule manager 18 determines, based on the result of a measurement of, or of an estimation of expansion time of the intermediate code image data made by the measurement unit 20, whether or not the intermediate code image data in each of the bands can be expanded to raster data in real-time. Here, in a case where there exists intermediate code image data which cannot be expanded in real-time, the schedule manager 18 starts the overlapped graphics processor 14 to execute the overlapped graphics processing on a band including the intermediate code image data which cannot be expanded in real-time. Thereby, overlapping between image areas represented by intermediate code image data can be eliminated, and real-time expansion processing is thus made possible by preventing intermediate code image data from being unnecessarily written into the band buffers 26 during the expansion processing.

[0047] (Overview of Overlapped Graphics Processing) Next, the contents of the overlapped graphics processing to be executed by the overlapped graphics processor 14 of the present embodiment will be explained with reference to Fig. 2. It should be noted that the description of the overlapped graphics processing is given of a case where multiple objects overlapped with one another are four graphics denoted by reference numerals A, B, C and D in Fig. 3. For this reason, the term "graphics" are used as the same meaning as objects.

[0048] The overlapped graphics processing in this embodiment is the same as that proposed in Japanese Patent Application No. Hei 8-276652, and is executed on each object in the reversed sequence of a rendering sequence. For example, in Fig. 2, the graphics are processed in the sequence of D, C, B and A, which is the reversed

sequence of the rendering sequence.

[0049] Moreover, the overlapped graphics processing includes an extraction process, a creation process and an updating process. In the extraction process, an overlapped area which is an overlapped area of a certain object with the area obtained by combining areas of all other objects to be rendered after the certain object is extracted. In the creation process, edge lists of areas obtained by removing the extracted overlapped area from the certain object are created in a working memory of the list storage unit 11. In the updating process, the edge lists of the objects stored in a saving memory of the list storage unit 11 are updated by the created edge lists.

[0050] Specifically, in Fig. 2, the overlapped area of the graphic C and the outline of the graphic D (MASK(D)) to be rendered after the graphic C is extracted. Then, an edge list of an area obtained by removing the overlapped area from the graphic C (Cand to MASK(D)) is created. Subsequently, the edge list of the graphic C is updated by the edge list of the created area (Cand to MASK(D)).

[0051] At this point, the edge list of the graphic C is fixed since the aforementioned area (Cand to MASK(D)) is not overwritten by the graphics B and A which are subjected to the overlapped graphics processing later. Thus, the edge list of the graphic C is updated by the fixed edge list of the graphic C. Thereby, since the edge list created in the working memory is no longer necessary after the completion of the updating process, the working memory in which the edge list is stored is released, and thus, the working memory can be reused for another process.

[0052] Likewise, the overlapped area of the graphic B and the outline of the graphic D or C (MASK(DorC)) to be rendered after the graphic B is extracted. Then, the edge list of the area obtained by removing the overlapped area from the graphic B (Band to MASK(DorC)) is created. Subsequently, the edge list of the graphic B is updated by the edge list of the created area (Band to MASK(DorC)).

[0053] Furthermore, the overlapped area of the graphic A and the outline of the

graphic D, C or B (MASK(DorCorB)) to be rendered after the graphic A is extracted. Then, the edge list of the area obtained by removing the overlapped area from the graphic A (Aand to MASK(DorCorB)) is created. Subsequently, the edge list of the graphic A is updated by the edge list of the created area (Aand to MASK(DorCorB)).

[0054] It should be noted that, the extraction process in the aforementioned overlapped graphics processing is characterized in that, as the data representing the area obtained by combining all of the graphics to be rendered after the target graphic, outline data, for example, the outline data of the graphic D (MASK(D)), the outline data of the graphic D or C (MASK(DorC)), or the like are used. Such outline data only include information related to the outline of the area, and do not include various information related to rendering such as a line type or a color value unlike edge list. Thus, the amount of data of the outline data is very small. By use of the outline data having a small amount of data in such extraction process, it is possible to suppress an amount of usage of the working memory in the extraction process to be small.

[0055] (Detailed Description of Overlapped Graphics Processing) Next, the overlapped graphics processing will be explained in detail along the flowchart of Fig. 6.

[0056] It should be noted that an individual data block constituting an edge list before updating each graphic is termed as an original graphic edge, and an individual data block constituting an edge list after updating each graphic is termed as a new graphic edge. Furthermore, the outline of an area obtained by combining all the graphics to be rendered after a target graphic is processed is termed as a mask. As in the case of graphics, this mask is also represented by an edge list, and an individual data block constituting an edge list of a mask is termed as a mask edge. Incidentally, the outline (mask) of the graphic D is expressed as MASK(D), and the outline (mask) of the graphic D or C is expressed as MASK(DorC) in Fig. 2. These edge list and mask can be constituted of five types of data storage areas, which are of: an attribute; a starting

point; an ending point; a run length; and a pointer to the next mask or edge list as shown in Fig. 4. Among these, an edge list can be identified when the attribute data is "0," and a mask can be identified when the attribute data is "1."

[0057] In Step 101 of Fig 6, as a processing range, one page is divided into multiple bands along the sub scanning direction, and then the plurality of bands are set. In next Step 102, the edge list of the target object is read, and the range of a Y coordinate of the graphic is detected (Y_S to Y_E). Then, in next Step 103, whether or not the target object exists in the bands being set is determined.

[0058] Here, in a case where the target object exists in the bands, the process proceeds to Step 104, and Y_S is set as the Y coordinate of the scanning line which is the target to be processed. Hereinafter, the scanning line which is the target to be processed is termed as a line Y. In next Step 106, (the X coordinate of the starting point, the X coordinate of the ending point) of the first original graphic edge in the line Y are set to (SX, EX), respectively. In Step 106, (the X coordinate of the starting point, the X coordinate of the ending point) of the first mask edge in the line Y are also set to (MSX, MEX), respectively. It should be noted that the original graphic edge and the mask edge are set as the processing targets in the line Y sequentially in the main scanning line direction, here.

[0059] It should be noted in the description below that (SX, EX) respectively indicate (the X coordinate of the starting point, the X coordinate of the ending point) of the first original graphic edge in the line Y. (MSX, MEX) respectively indicate (the X coordinate of the starting point, the X coordinate of the ending point) of the mask edge in the line Y. (NSX, NEX) respectively indicate (the X coordinate of the starting point,

the X coordinate of the ending point) of the next original graphic edge in the line Y. (NMSX, NMEX) respectively indicate (the X coordinate of the starting point, the X coordinate of the ending point) of the next mask in the line Y.

[0060] Moreover, the following steps include processes to compare aforementioned various values of the X coordinate with one another. Figs. 5(A) to (F) are used for the purpose of describing such processes. In the Figs. 5(A) to (F), the direction from left to right corresponds to the main scanning direction, and mask edges (MSX to MEX) prior to comparison processes and original graphic edges (SX to EX) prior to comparison process are shown above the arrows each pointing downward. Moreover, mask edges (MSX to MEX) after the comparison processes and original graphic edges (SX to EX) after the comparison processes are shown below the arrows each pointing downward in the Figs. 5(A) to (F).

[0061] In next Step 108 in Fig. 6, whether or not an MSX is greater than an EX is determined. In a case where the MSX is greater than the EX as shown in Fig. 5(A), the process proceeds to Step 110, and the subroutine A of Fig. 7 is executed. In the subroutine A, (SX, EX) are added as a new graphic edge (Step 202). In a case where (MSX-1) is equal to the EX, since the original graphic edge and the mask edge can be incorporated into one mask edge, (SX, MEX) are set to (MSX, MEX) (Step 208). In a case where (MSX-1) is not equal to the EX, (SX, EX) are added to the mask edge (Step 206). Then, the next original graphic edge is set to (NSX, NEX) (Step 210), and the process returns to the main routine of Fig. 6. Then, the process proceeds to Step 120 to be described later.

[0062] In a case where the MSX is not greater than the EX in aforementioned Step 108, the process proceeds to Step 112, and whether or

not (MSX ≥ SX), (MEX > EX) and (MSX ≤ EX) are true is determined. In a case where (MSX ≥ SX), (MEX > EX) and (MSX ≤ EX) are true as shown in Fig. 5(B), the process proceeds to Step 114, and then, executes the subroutine B in Fig. 8. In the subroutine B, (SX, MSX-1) are added as a new graphic edge (Step 222), and (SX, MEX) are set to (MSX, MEX) (Step 224). Then, in the subroutine B, the next original graphic edge is set to (NSX, NEX), and the process returns to the main routine in Fig. 6, and then, proceeds to Step 120 to be described later.

[0063] In a case where a negative determination is made in aforementioned Step 112, the process proceeds to Step 116, and whether or not (MSX ≤ SX) and (MEX ≥ EX) are true is determined. In a case where (MSX ≤ SX) and (MEX ≥ EX) are true as shown in Fig. 5(C), the process proceeds to Step 118, and then the next original graphic edge is set to (NSX, NEX), and proceeds to Step 120. In Step 120, (NSX, NEX) are set to (SX, EX), and the process proceeds to Step 136 to be described later.

[0064] In a case where a negative determination is made in aforementioned Step 116, the process proceeds to Step 122, and whether or not (MSX ≥ SX) and (MEX ≤ EX) are true is determined. In a case where (MSX ≥ SX) and (MEX ≤ EX) are true as shown in Fig. 5(D), the process proceeds to Step 124, and then executes the subroutine D in Fig. 9. In the subroutine D, (SX, MSX-1) are added as a new graphic edge (Step 242), and (SX, EX) are set to (MSX, MEX) (Step 244). Moreover, (MEX+1, EX) are set to (NSX, NEX) (Step 246). Furthermore, the next mask edge is set to (NMSX, NMEX) (Step 248), and then the process returns to the main routine in Fig. 6, and proceeds to Step 134 to be described later.

[0065] In a case where a negative determination is made in aforementioned Step 122, the process proceeds to Step 126, and whether or not (MSX ≤ SX),

(MEX < EX) and (MEX ≥ SX) are true is determined. In a case where (MSX ≤ SX), (MEX < EX) and (MEX ≥ SX) are true as shown in Fig. 5(E), the process proceeds to Step 128, and then executes the subroutine E in Fig. 10. In the subroutine E, (MEX+1, EX) are set to (SX, EX) (Step 262), and (MSX, EX) are set to (MSX, MEX) (Step 264). Then, in the subroutine E, the next mask edge is set to (NMSX, NMEX) (Step 266), and then the process returns to the main routine in Fig. 6, and proceeds to Step 134 to be described later.

[0066] In a case where a negative determination is made in aforementioned Step 126, the process proceeds to Step 130, and then, whether or not the MEX is smaller than the SX is determined. In a case where the MEX is smaller than the SX as shown in Fig. 5(F), the process proceeds to Step 132, and then executes the subroutine F in Fig. 11. In the subroutine F, (SX, EX) are added as a new graphic edge (Step 282). Here, in a case where (MEX+1) is equal to the SX, since the original graphic edge and the mask edge can be incorporated into one mask edge, (MSX, EX) are set to (MSX, MEX) (Step 288). In a case where (MEX+1) is not equal to the SX, (SX, EX) are added to the mask edge (Step 286). Then, the next mask edge is set to (NMSX, NMEX) (Step 290), and then, the process returns to the main routine in Fig. 6, and proceeds to Step 134. In Step 134, after (MSX, MEX) are registered as the mask edge, (NMSX, NMEX) are set to (MSX, MEX), and then the process proceeds to next Step 136.

[0067] In Step 136, whether or not (SX, EX) are blank data (the next original graphic edge is not present in the line Y) is determined. In next Step 140, whether or not (MSX, MEX) are blank data (the next mask edge is not present in the line Y) is determined. Here, in a case where both (SX, EX) and (MSX, MEX) are not blank data, since there remain data yet to be

compared, the process returns to Step 108.

[0068] In a case where (SX, EX) are blank data in Step 136, since the next original graphic edge is not present in the line Y, the process proceeds to Step 138, and then the edge list of the mask at that point of time is stored in the list storage unit 11. It should be noted that, since the edge list of the mask is data related to the outline only, and since the amount of data is very small, an area of the list storage unit 11 to be used can be small. On the other hand, in a case where (MSX, MEX) are blank data in Step 140, since the next mask edge is not present in the line Y, the process proceeds to Step 142. Then, after a remaining original graphic edge is added as the mask edge, the edge list of the mask at that point of time is stored in the list storage unit 11. As in the aforementioned case, since the edge list of the mask is data related to the outline only, and since the amount of data is very small, an area of the list storage unit 11 to be used can be small.

[0069] After the execution of the processing in Steps 138 and 142, whether or not the Y coordinate of the line Y is equal to Y_E is determined in Step 144. When the processing on one graphic which is the processing target is not yet completed, the Y coordinate of the line Y is smaller than Y_E. Thus, a negative determination is made in Step 144, and then the Y coordinate of the line Y is incremented by one in Step 146. Thereafter, the process returns to Step 106, and then executes the processing in Steps 106 to 142 with a new line Y as the target.

[0070] As described above, a single graphic which is a processing target is processed for each scanning line, and then the edge list of the original graphic edge is updated by the edge list of a new graphic edge. When the processing on a single graphic is completed by completing the processing on the line Y, in which the Y coordinate is equal to the Y_E, as the target, an

affirmative determination is made in Step 144, and then the process proceeds to Step 147. The edge list of the original graphic being stored in the list storage unit 11 is updated by the edge list of the new graphic edge in Step 147. Then, whether or not the processing on all of the graphics is completed is determined in Step 148.

[0071] In a case where there remains a graphic which has not been subjected to the processing, the process returns to Step 102, and then executes the processing in Steps 102 to 147 with a new graphic as the target. Then, when the processing on all of the graphics is completed, the overlapped graphics processing is ended.

[0072] (Operation of First Embodiment) Next, as an operation of the first embodiment, an intermediate code image data reconstruction process will be explained. In the intermediate code image data reconstruction process, edge lists created by the aforementioned overlapped graphics processing are scanned along a predetermined direction, and thereby intermediate code image data under each of the objects are reconstructed from the edge lists. The intermediate code image data reconstruction process is executed by the upper level processor 12 at the time of printing out.

[0073] In Step 302 in Fig. 12, edge lists stored in the list storage unit 11 are read out, and one of the edge lists is set as the processing target. In next Steps 304 and 306, whether or not the object represented by the target edge list is an image, and whether or not the object represented by the target edge list is a color graphic, are respectively determined. The process proceeds to Step 308, when the target object is an image, and proceeds to Step 312 when the target object is a color graphics. When the target object is other than color graphics (for example, font or black and while

graphics), the process proceeds to Step 310.

[0074] Among these steps, in Step 308, the subroutine of an intermediate code image data reconstruction process 1 shown in Fig. 13 is executed. In this intermediate code image data reconstruction process 1, edge lists are scanned along the main scanning line direction (the X axis direction in a two dimensional coordinate system), and thereby the intermediate code image data are reconstructed from the edge lists.

[0075] Specifically, in Step 332 in Fig. 13, whether or not an unprocessed edge list exists in a target scanning line (the top scanning line at first) is determined. When there exists an unprocessed edge list, the scanning of the edge lists (Step 334) and the reconstruction of the intermediate code image data on the basis of the information obtained by the scanning are executed (Step 336) in the sequence along the main scanning direction with respect to the edge lists on the scanning line.

[0076] Then, upon completion of the aforementioned processing on all of the edge lists on the scanning line, the next scanning line is set as the target scanning line in Step 340, and then the processing in Steps 332 to 336 on the target scanning line being set is executed.

[0077] As described above, by executing the processing in Steps 332 to 336 for each of the scanning lines, the edge lists are scanned along the main scanning direction in the sequence, for example, of (1), (2), (3), and so on shown in Fig. 16(A), and thus the intermediate code image data are reconstructed.

[0078] In a case where the type of object is image as described above, the intermediate code image data are merely retained in a state of including clip information, and the entity of the images exists in a different memory area being as it is, or being compressed. For this reason, in the expansion

process, the data are read out in the sequence of addresses from the area where the entity of the image is retained, and then, need to be written into memory by clipping with the intermediate code image data (=the set of edge lists) being sorted in the main scanning line direction and the sub scanning line direction. Thus, it is desirable that the edge lists be scanned with the main scanning line direction as its scanning direction as described above.

[0079] On the other hand, when the process proceeds to Step 310 in a case where the object is a font or a black and white graphics, the subroutine of an intermediate code image data reconstruction process 2 shown in Fig. 14 is executed. In this intermediate code image data reconstruction process 2, the edge lists are scanned by switching the scanning direction to the main scanning line direction or the sub scanning line direction (the Y axis direction in the two dimensional coordinate system) in accordance with a distance between the edge list of the scanning target and the edge list to become the next scanning target. Thereby, the intermediate code image data are reconstructed from the edge lists.

[0080] Specifically, in Steps 352 and 354 in Fig. 14, the scanning of the first edge list, and the reconstruction of the intermediate code image data on the basis of the information obtained from the scanning are executed, respectively. In next Step 356, whether or not the number of unprocessed edge lists is equal to or greater than two is determined.

[0081] Here, when the number of unprocessed edge lists is equal to or greater than two, the process proceeds to Step 358, and the distance to the edge list which is to become the next scanning target when viewed in the main scanning direction and the distance to the edge list which is to become the next scanning target when viewed in the sub scanning direction are compared. In next Step 360, based on the result of comparison, the

direction towards the edge list with the shorter distance is set as the scanning direction. Then, in next Step 362, the next unprocessed edge list when viewed in the aforementioned scanning direction being set is set to be the target edge list. Moreover, in next Steps 364 and 366, the scanning of the target edge list, and the reconstruction of the intermediate code image data on the basis of the information obtained by the scanning are executed, respectively.

[0082] As described above, the intermediate code image data are reconstructed by scanning the edge lists by switching the scanning direction to the main scanning line direction or the sub scanning line direction in accordance with the distance of the edge list of the scanning target and the edge list to become the next scanning target.

[0083] Thereby, the edge lists are scanned in the sequence of (1), (2), (3), and so on as shown in Fig. 16(B), and the intermediate code image data are reconstructed.

[0084] Incidentally, in aforementioned Steps 358 to 362, the processing may be performed in the following manner. For example, one of the intermediate code image data formats, which represent an edge list, is defined using two digits of data type, three digits of run length, one digit of the difference in the Y direction, and two digits of the difference in the X direction as shown in Fig. 17(B).

[0085] Here, in Fig. 17(A), when the current scanning target is to be an edge list (1), there exist two edge lists (2) and (3) as the next scanning targets. Here, in a case where the difference DX' between the edge list (1) and the edge list (2) in the X direction is equal to or less than 3 (= the difference DX' is within two digits), the next scanning target is to be the edge list (2). On the other hand, in a case where the difference DX' in the

X direction of the edge list (2) is greater than 3 (=the difference DX' is equal to or greater than 3), where the difference DX'' in the X direction of the edge list (3) is equal to or less than 3 (=the difference DX' is within two digits), and where difference DY' in the Y direction is equal to or less than 1 (=the difference DY' is within a single digit), the next scanning target is to be the edge list (3). Thereby, the amount of memory usage for storing the intermediate code image data can be reduced.

[0086] The processing in the aforementioned Steps 358 to 366 is continued until the number of unprocessed edge lists becomes one. Then, when the number of unprocessed edge lists becomes one, the scanning of the last edge list, and the reconstruction of intermediate code image data on the basis of information obtained by the scanning in Steps 368 and 370 are executed, respectively, and then the processing is ended.

[0087] As described above, in a case where the type of object is font or black and white graphics, there may be a possibility that the shape thereof after the overlapped portions are removed become quite complicated. Thus, it is desirable that the edge lists be scanned by switching the scanning direction to the main scanning line direction or the sub scanning line direction for each of the edge lists in accordance with the distance to the edge list which may become the next scanning target. By scanning in the direction towards the edge list with the shorter distance as the scanning direction, the amount of data of the distance information between the edge list and an adjacent edge list can be reduced. Thus, the memory size for storing the data can be reduced, and thereby the memory space can be saved.

[0088] Furthermore, in a case where the object is color graphics, the process proceeds to Step 312, and then, the subroutine of an intermediate

code image data reconstruction process 3 shown in Fig. 15 is executed. In this intermediate code image data reconstruction process 3, edge lists are scanned along the sub scanning direction, and thereby the intermediate code image data are reconstructed from the edge lists.

[0089] Specifically, in Step 382 in Fig. 15, whether or not an unprocessed edge list exists in the target scanning line (the top scanning line at first) is determined. When an unprocessed edge list exists, the scanning of the edge list (Step 384), and the reconstruction of intermediate code image data on the basis of the information obtained by the scanning are executed (Step 386), for a first edge list on the scanning line when viewed in the main scanning direction.

[0090] Then, in next Step 388, whether or not the target scanning line is the last scanning line is determined. In a case where the target scanning line is not the last scanning line, the next scanning line is set as the target scanning line in Step 390, and thereafter, the process returns to Step 382, and then the processing in Steps 382 to 386 is executed. However, in a case where there does not exist an unprocessed edge list in the target scanning line, the next scanning line is set as the target scanning line in Step 396 after confirming that the processing is not yet completed for all of the scanning lines in Step 394, and then the process returns to Step 382.

[0091] As described above, only for the first unprocessed edge list in all of the scanning lines, the scanning of the edge lists (Step 384) and the reconstruction of intermediate code image data based on the information obtained by the scanning are executed (Step 386). Thereby, the edge lists are scanned along the sub scanning direction.

[0092] Furthermore, after the processing on the first unprocessed edge list in each of the scanning lines is completed, the first scanning line is set as

the target scanning line in Step 392, and then the process returns to Step 382. Then, a second unprocessed edge list in each of the scanning lines is executed.

[0093] As described above, the edge lists are scanned along the sub scanning direction in the sequence, for example, of (1), (2), (3), and so on shown in Fig. 16 (C), and thereby the intermediate code image data are reconstructed.

[0094] In a case where the type of object is color graphics, differently from black and white graphics, the possibility of the shape thereof to become rectangular is high even when the overlapped portions with other objects are removed. Thus, by scanning the edge lists in the sub scanning line direction as the scanning direction, the edge lists can be reconstructed as the intermediate code image data having a small amount of data such as rectangular display lists or codes using functions to represent the continuity for example. Thereby, the size of memory for storing data can be reduced, and the memory can be thus saved.

[0095] In the main routine in Fig. 12, in accordance with the type of each of the objects, any one of the intermediate code image data reconstruction processes 1 to 3, which have been described above is executed on each of the objects. Then, the intermediate code image data reconstruction process is completed for all of the objects. In a case where there are no any other bands which need to be subjected to the overlapped graphics processing (in a case where a negative determination is made in Step 315 of Fig. 12), the process proceeds to Step 316. Then, the intermediate code image data reconstructed in Steps 308, 310 and 312 are converted into raster data. In next Step 318, the raster data are printed out. Incidentally, in a case where there exists another band which needs to be subjected to the

overlapped graphics processing in Step 315, the process returns to Step 302, and then the processing after the processing in Step 302 is executed on the band.

[0096] According to the first embodiment described above, updated edge lists for each of the objects are scanned, in accordance with the type of objects, by switching any one of the following modes: one in which the scanning direction is set to be the main scanning line direction, one in which the scanning direction is set to be the sub scanning line direction, and one in which the scanning direction is switched between the main scanning line direction and the sub scanning line direction for each of the edge lists in accordance with predetermined switching conditions. Then, the intermediate code image data are reconstructed from the edge lists for each of the objects. Thereby, the data amount of the intermediate code image data can be reduced. Thus, the amount of memory usage can be reduced while the expansion processing speed to raster data at the time of an output is improved.

[0097] Incidentally, in the aforementioned intermediate code image data reconstruction process 3, in order to attempt to increase the processing speed, a counter table 70 of bandwidths which hold the number of edge lists for the respective scanning lines, and a pointer table 80 of bandwidths which hold pointers to the top edge list for each of the scanning lines, shown in each of Figs. 18(A) and (B), may be employed.

[0098] In Figs. 18(A) and (B), the contents of the counter table 70 and the pointer table 80 for before scanning in the sub scanning line direction once, and the contents of the counter table 70 and the pointer table 80 for after scanning in the sub scanning line direction once are respectively shown.

[0099] First, a scanning line which is not 0 is found by referring to the

counter table 70, and then, in a case where the content of the counter table 70 is not 0, by referring to the pointer table 80 of the scanning line, the pointer to the top edge list is obtained.

[0100] It should be noted that, although the address of the pointer table 80 in line 2 of Fig. 18(A) is a pointer to a mask, this is because the mask pointed and indicated by the pointer table 80 and the edge list indicated and pointed by the mask are adjacent to each other. Furthermore, although the edge list which is no longer needed after the scanning of each of the edge lists is converted into a mask for the next object, in a case where the edge list is adjacent to the masks immediately before and after the edge list, they are incorporated into one mask.

[0101] The edge list which has been scanned is converted into a mask. Then, the value of the counter table 70 is subtracted by one, and in a case where the result obtained by subtracting one is not 0, the pointer to the next edge list on the same scanning line is found. Then, the value of the pointer table 80 is updated by this pointer value (however, in a case where the mask immediately before the edge list and the edge list are adjacent to each other, the pointer to the mask immediately before the edge list is set to the pointer table 80).

[0102] Likewise, the processing is performed on the next scanning line, and by repeating the processing until the last scanning line, one time of the scanning processing in the sub scanning line direction is completed.

[0103] It should be noted that the minimum value and the maximum value (the effective scanning line widths respectively shown in Figs. 18(A) and (B)) of the scanning lines in which the edge lists constituting the target object of the overlapped graphics processing exist are held. Thus, only the counter table 70 within the range of the scanning lines may be accessed.

Thereby, the processing speed of the scanning can be further increased.

[0104] Moreover, although the aforementioned embodiment shows an example of employing the overlapped graphics processing proposed in Japanese Patent Application No. Hei 8-276652 as the overlapped graphics processing, the overlapped graphics processing is not limited to this, and another overlapped graphics processing may be employed.

[0105] [Second Embodiment] Next, a second embodiment which corresponds to the invention of claims 4 to 7 as recited in the scope of claims will be explained. It should be noted that, since the configuration of the image processing apparatus in the second embodiment is the same as the one used in the first embodiment, the description thereof is omitted.

[0106] (Description of Segmentalization of Objects) Fig. 19 shows a case where a single object 50 is overwritten by multiple objects 60. Here, in a case where both of the objects 60 of the upper layer and the object 50 of the lower layer are rectangular graphics (however, the colors of these are different), and where the graphic is represented by rectangular intermediate code image data format (provided that both of the height and width thereof are within the range of the rectangular intermediate code image data format), the graphic of the lower layer is represented by a single piece of the rectangular intermediate code image data, and the graphics of the upper layer are represented by 23 pieces of the rectangular intermediate code image data (total of 24 pieces).

[0107] Next, considered is a case where the overlapped graphics processing described in the first embodiment is executed on bands in which the aforementioned graphics exist.

[0108] Fig. 20 shows a state immediately after the aforementioned overlapped graphics processing is executed on the object 50 of the lower.

As it is apparent from this Fig. 20, the objects, which are the targets for the overlapped graphics processing, are on upper layer higher than the objects of the lower layer and become masks 62, and portions of the object which does not overlap with the masks 62 become the objects 52 of the lower layer after the overlapped portions are removed. Thereby, 23 pieces of the objects of the upper layer and 24 pieces of the objects of the lower layer become a total of 47 pieces of the rectangular intermediate code image data. Thus, the amount of memory usage of the intermediate code image data increases approximately twice by the overlapped graphics processing.

[0109] It should be noted that the processing time is not increased since the amount of unnecessary writing of the overlapped portions to the memory in the expansion process is reduced. However, there remains a problem that the amount of memory usage for the intermediate code image data increases.

[0110] Furthermore, supposing a case where an object of the lower layer is an image, the portions segmentalized by the objects of the upper layer cannot be represented by rectangular intermediate code image data (since the intermediate code image data which are clip information are not sorted in the main scanning line direction and the sub scanning line direction, the image cannot be accessed in the sequence of addresses at high speed. Thus, the image needs to be read by the addresses not in the stepping manner, or read at the amount of rectangular intermediate code image data equivalent to that of the image which are read in the sequence of addresses). Thus, intermediate code image data become of a set of edge lists.

[0111] Here, the number of edge lists becomes 24 pieces in the main scanning line direction, and the number of edge lists becomes the number of pieces of the number of scanning lines equivalent to the height of the object.

For example, assuming that the height is equivalent to 40 scanning lines, the total number of edge lists is 960 pieces (=24 × 40). Thus, the intermediate code image data represented by a single piece of the rectangular intermediate code image data before the overlapped graphics processing becomes intermediate code image data having 960 pieces of edge lists after the overlapped graphics processing. Accordingly, the amount of memory usage significantly increases. Moreover, although the image can be accessed in the sequence of addresses at high speed, eventually, the image clipped by the individual edge list is to be written into the memory. Accordingly, the expansion processing time becomes longer.

[0112] For this reason, in the second embodiment, in a case where the type of object is not an appropriate type for the overlapped graphics processing (specifically, images), the object is controlled not to be subjected to the overlapped graphics processing. In the operation to be described below, such a control process will be explained.

[0113] (Operation of Second Embodiment) Hereinafter, in only a case where the type of object is image, the process to control not to perform the overlapped graphics process on the object will be explained along the flowchart in Fig. 23.

[0114] In Step 402 of Fig. 23, as a processing range, multiple bands divided from one page along the sub scanning direction are set. In next Step 404, edge lists of the target objects are read out, and the range of the Y coordinate of the object (Y_S to Y_E) is detected. Then, in next Step 406, whether or not the target objects exist in the bands being set is determined.

[0115] Here, in a case where the target objects exist in the bands, the process proceeds to Step 408, and whether or not the target object is an

image is determined. Here, in a case the target object is other than an image, the process proceeds to Step 410, and then, the aforementioned overlapped graphics processing is executed. On the other hand, in a case where the target object is other than an image, the process proceeds to Step 412, and in the same manner as the aforementioned overlapped graphics processing, mask edge lists are created, and then stored in the list storage unit 11. Specifically, the mask edge lists corresponding to the masks 64 shown in Fig. 21 are created, and then saved.

[0116] Subsequently, in next Step 414, intermediate code image data (the intermediate code image data from which the overlapped portions are not removed) are temporarily retained in a retaining block in a working area of the list storage unit 11.

[0117] It should be noted that, as to an image, since the format of the original intermediate code image data is retained, the intermediate code image data are retained in the top of an intermediate code image data retaining block corresponding to bands. Thus, the image is rendered prior to an object other than an image, and an upper and lower relationship of the objects in the image to be printed out is secured. In other words, by rendering an image after an object other than the image, it is possible to prevent the object to be on the layer higher than the image from being overwritten by the image in advance.

[0118] Thereafter, the processing in Steps 404 to 414 is executed on each of the target objects which exist in the bands. Then, upon completion of the processing on all of the target objects which exist in the bands, the process proceeds to Step 418. Then, in only a case where the object is an image (that is, in a case of an object which is not to be subjected to the overlapped graphics processing), as shown in Fig. 22, the rendering

sequence is changed, and also as shown in Fig. 24(B), intermediate code image data 96 of the image retained in the retaining block in the working area are retained in an official retaining block in the list storage unit 11 before intermediate code image data 98 of an object other than an image (incidentally, in a case where the intermediate code image data of the image are subjected to the overlapped graphics processing, as shown in Fig. 24(A), there is a case where the intermediate code image data 98 of an object other than an image are retained before the intermediate code image data 96 of the image which have been subjected to the overlapped graphics processing).

[0119] Thereafter, at the point when the processing in Steps 402 to 418 is competed for the objects in all of the bands, the processing of Fig. 23 is ended.

[0120] As described above, an object (image) which is not appropriate for the overlapped graphics processing to be executed is controlled not to be subjected to the overlapped graphic processing. Thus, it is possible to prevent an increase in an amount of memory usage for intermediate code image data and also a reduction in the expansion processing speed due to segmentalization of objects.

[0121] Furthermore, the intermediate code image data of an image from which the overlapped portions are not removed are temporarily retained in the retaining block in a working area. Then, after the processing on all of the target objects in the bands is completed, the rendering sequence of the intermediate code image data is changed as shown in Fig. 22. Then, the intermediate code image data retained in the retaining block in the working area are retained in the official retaining block. Thus, even in a case where multiple images exist in one band, supposedly, the image on the

upper layer side is not overwritten by the image on the lower layer side, and the rendering sequence can be normally retained.

[0122] It should be noted that, in a case where multiple images which are not subjected to the overlapped graphics processing exist (for example, images in the aforementioned case), the overlapped graphics processing may be performed among only these objects. In a case where such overlapped graphics processing among the objects which have not been subjected to the overlapped graphics processing normally ends, the rewriting process of the intermediate code image data after changing the rendering sequence thereof as shown in Fig. 22 is no longer necessary. Thus, the processing efficiency can be improved.

[0123] Moreover, as a condition of not performing the overlapped graphics processing, in addition to the type of object, the degree of the overlapped state between the target object and another object can be cited as an example. For example, in a case where the degree of overlapped state is one as shown in Fig. 19, the number of times that each of the edge lists is cut by a mask is counted. Then, when the counted value is equal to or greater than a certain value, it is considered that segmentalization has occurred. Thus, it is desirable that the object be controlled not to be subjected to the overlapped graphics processing.

[0124] As in the manner described above, by prohibiting an object having the degree of the overlapped state with another object at a level equal to or greater than a predetermined level from being subjected to the overlapped graphics processing, it is possible to prevent an increase in an amount of memory usage of intermediate code image data and a reduction in an expansion processing speed due to the segmentalization of the object from occurring in advance.

[0125]

[Effect of the Invention] As has been described above, according to the invention as recited in claims 1 to 4, intermediate code image data are reconstructed by scanning updated edge lists along an appropriate scanning direction. Thus, an amount of data of the intermediate code image data can be reduced, and the expansion processing speed to raster data at the time of output can be improved while reducing an amount of memory usage.

[0126] Furthermore, according to the invention as recited in claim 5, the overlapped graphics processing is prohibited from being performed on an object for which the execution of the overlapped graphic processing is not appropriate. Thus, it is possible to prevent an increase in an amount of memory usage of intermediate code image data and a reduction in the expansion processing speed due to the segmentalization of the object.

[0127] Moreover, according to the invention as recited in claim 6, the overlapped graphics processing is prohibited from being performed on an object having the degree of the overlapped state with another object at a level equal or greater than a predetermined level. Thus, it is possible to prevent an increase in an amount of memory usage of intermediate code image data and a reduction in the expansion processing speed due to the segmentalization of the object from occurring in advance.

[0128] Still furthermore, according to the invention as recited in claim 7, in a case where there exist multiple objects for which the overlapped graphics processing is prohibited, the overlapped graphics processing is controlled to be executed only among the objects for which the overlapped graphics processing is prohibited. Thereby, it is no longer necessary to change the rendering sequence of intermediate code image data, and then to store the intermediate code image data in an official memory again after the

intermediate code image data are temporarily stored in a working area of memory. Thus, the processing efficiency can be improved. Moreover, particularly, in a case where an object is an image, it is possible to reduce not only the intermediate code image data which retain clip information, but also source data.

[Brief Description of the Drawings]

[Fig. 1] Fig. 1 is a functional block diagram of an image processing apparatus according to first and second embodiments.

[Fig. 2] Fig. 2 shows diagrams indicating an overview of overlapped graphic processing.

[Fig. 3] Fig. 3 is a diagram showing a rendering area on which multiple graphics (objects) are written in an overlapped manner.

[Fig. 4] Fig. 4 shows diagrams indicating structures of an edge list and a mask.

[Figs. 5] Figs. 5(A), (B), (C), (D), (E) and (F) are diagrams for the purpose of describing processes to compare the coordinates between mask edges and original graphic edges respectively in cases where: (MSX > EX) in (A); (MSX $\geq$ SX), (MEX > EX) and (MSX $\leq$ EX), in (B); (MSX $\leq$ SX) and (MEX $\geq$ EX) in (C); (MSX $\geq$ SX) and (MEX $\leq$ EX) in (D); (MSX $\leq$ SX), (MEX < EX) and (MEX $\geq$ SX) in (E); and (MEX < SX) in (F).

[Fig. 6] Fig. 6 is a flowchart showing a main routine of the overlapped graphics processing.

[Fig. 7] Fig. 7 is a flowchart showing a subroutine A.

[Fig. 8] Fig. 8 is a flowchart showing a subroutine B.

[Fig. 9] Fig. 9 is a flowchart showing a subroutine C.

[Fig. 10] Fig. 10 is a flowchart showing a subroutine D.

[Fig. 11] Fig. 11 is a flowchart showing a subroutine F.

[Fig. 12] Fig. 12 is a flowchart showing the main routine in the first embodiment.

[Fig. 13] Fig. 13 is a flowchart showing a subroutine of an intermediate code image data reconstruction process 1.

[Fig. 14] Fig. 14 is a flowchart showing a subroutine of an intermediate code image data reconstruction process 2.

[Fig. 15] Fig. 15 is a flowchart showing a subroutine of an intermediate code image data reconstruction process 3.

[Figs. 16] Figs. 16(A), (B) and (C) are diagrams showing the scanning sequences of edge lists in the intermediate code image data reconstruction processes 1, 2 and 3, respectively.

[Figs. 17] Figs. 17(A) is a diagram for the purpose of explaining the process to determine the scanning direction of edge lists based on the distance between the edge lists in the intermediate code image data reconstruction process 2. Fig. 17(B) is a diagram showing an example of an intermediate code image data format representing an edge list.

[Figs. 18] Figs. 18(A) and (B) are diagrams each showing a count table and a pointer table used for scanning edge lists in the intermediate code image data reconstruction process 3 at high speed. Fig. 18(A) is a diagram showing a count table and a pointer table prior to the scanning, and Fig. 18(B) is a diagram showing a count table and a pointer table after the scanning is performed once.

[Fig. 19] Fig. 19 is a diagram showing a state in which a single object is overwritten by multiple objects.

[Fig. 20] Fig. 20 is a diagram showing a state in which the object of the lower layer is segmentalized by the overlapped graphics processing.

[Fig. 21] Fig. 21 is a diagram showing a state in which an object (an

image) not to be desirably subjected to the overlapped graphics processing is masked.

[Fig. 22] Fig. 22 is a diagram showing a method of retaining multiple intermediate code image data, in a case where there exist multiple intermediate code image data of objects (images) which are not desirably to be subjected to the overlapped graphics processing.

[Fig. 23] Fig. 23 is a flowchart showing a main routine in the second embodiment.

[Figs. 24] Fig. 24(A) is a diagram showing a conventional method of retaining intermediate code image data. Fig. 24(B) is a diagram showing a method of retaining intermediate code image data of an object (an image) which is not desirably subjected to the overlapped graphic processing according to the second embodiment.

[Explanation of Reference Numerals]

10: image processing apparatus;

11: list storage unit;

12: upper level processor;

14: overlapped graphics processor;

16: expansion unit;

26: band buffers;

30: printer engine.

[FIG. 1]

11 LIST STORAGE UNIT
12 UPPER LEVEL PROCESSOR
14 OVERLAPPED GRAPHICS PROCESSOR
16 EXPANSION UNIT
18 SCHEDULE MANAGER
20 MEASUREMENT UNIT
22 BAND BUFFER MANAGER
24 ERROR PROCESSOR
26 BAND BUFFERS
28 TRANSFER UNIT
30 PRINTER ENGINE
INPUT OF CODE IMAGE DATA
PRINT OUT
DATA FLOW
CONTROL INFORMATION FLOW

[FIG. 2]

ORIGINAL GRAPHICS
MASKS (SAVED GRAPHICS)
INITIAL VALUE (BLANK)
RESULTS

[FIG. 4]

EDGE LIST
MASK
ATTRIBUTE: 0 (EDGE LIST), 1 (MASK)
STARTING POINT
ENDING POINT
RUN LENGTH
POINTER TO NEXT MASK OR EDGE LIST

[FIG. 6]

START
101 SET BANDS
102 READ OUT EDGE LISTS OF ONE GRAPHIC (ORIGINAL GRAPHIC) IN

REVERSED SEQUENCE OF RENDERING SEQUENCE, AND DETECT RANGE IN Y COORDINATES (Y_S TO Y_E)

103 WHETHER OR NOT GRAPHIC EXISTS IN BANDS?

106 (SX, EX) = FIRST ORIGINAL GRAPHIC EDGE IN LINE Y
(MSX, MEX) = FIRST MASK EDGE IN LINE Y

110 SUBROUTINE A

112 WHETHER OR NOT (MSX ≥ SX), (MEX > EX) AND (MSX ≤ EX) ARE TRUE?

114 SUBROUTINE B

116 WHETHER OR NOT (MSX ≤ SX) AND (MEX ≥ EX) ARE TRUE?

118 (NSX, NEX) = NEXT ORIGINAL GRAPHIC EDGE

122 WHETHER OR NOT (MSX ≥ SX) AND (MEX ≤ EX) ARE TRUE?

124 SUBROUTINE D

126 WHETHER OR NOT (MSX ≤ SX), (MEX < EX) AND (MEX ≥ SX) ARE TRUE?

128 SUBROUTINE E

132 SUBROUTINE F

134 SET (NMSX, NMEX) AFTER REGISTERING (MSX, MEX)

136 WHETHER OR NOT (SX, EX) ARE BLANK DATA?

138 SAVE MASK EDGE LIST

140 WHETHER OR NOT (MSX, MEX) ARE BLANK DATA?

142 SAVE MASK EDGE LIST ON WHICH REMAINING ORIGINAL GRAPHIC EDGE LISTS ARE ADDED

147 UPDATE ORIGINAL GRAPHIC EDGE LIST BY NEW GRAPHIC EDGE LIST

148 WHETHER OR NOT ARE ALL GRAPHICS COMPLETED?

150 WHETHER OR NOT ARE ALL BANDS PROCESSED?
END

[FIG. 7]

SUBROUTINE A

202 ADD (SX, EX) AS NEW GRAPHIC EDGE

206 ADD (SX, EX) TO MASK EDGE

208 SET (SX, MEX) TO (MSX, MEX)

210 SET NEXT ORIGINAL GRAPHIC EDGE TO (NSX, NEX)
RETURN

[FIG. 8]

SUBROUTINE B
222 ADD (SX, MSX-1) AS NEW GRAPHIC EDGE
224 SET (SX, MEX) TO (MSX, MEX)
226 SET NEXT ORIGINAL GRAPHIC EDGE TO (NSX, NEX)
RETURN

[FIG. 9]

SUBROUTINE D
242 ADD (SX, MSX-1) AS NEW GRAPHIC
244 SET (SX, EX) TO (MSX, MEX)
246 SET (MEX+1, EX) TO (NSX, NEX)
248 SET NEXT MASK EDGE TO (NMSX, NMEX)
RETURN

[FIG. 10]

SUBROUTINE E
262 SET (MEX+1, EX) TO (SX, EX)
264 SET (MSX, EX) TO (MSX, MEX)
266 SET NEXT MASK EDGE TO (NMSX, NMEX)
RETURN

[FIG. 11]

SUBROUTINE F
282 ADD (SX, EX) AS NEW GRAPHIC
286 ADD (SX, EX) TO MASK EDGE
288 SET (MSX, EX) TO (MSX, MEX)
290 SET NEXT MASK EDGE TO (NMSX, NMEX)
RETURN

[FIG. 12]

START
302 READ OUT STORED EDGE LISTS
304 WHETHER OR NOT IS TARGET OBJECT IMAGE?
306 WHETHER OR NOT IS TARGET OBJECT COLOR GRAPHICS?

308 INTERMEDIATE CODE IMAGE DATA RECONSTRUCTION PROCESS 1
310 INTERMEDIATE CODE IMAGE DATA RECONSTRUCTION PROCESS 2
312 INTERMEDIATE CODE IMAGE DATA RECONSTRUCTION PROCESS 3
314 WHETHER OR NOT ARE ALL OBJECTS PROCESSED?
315 WHETHER OR NOT IS BAND FOR WHICH OVERLAPPED GRAPHICS PROCESSING IS NEEDED PRESENT?
316 CONVERT INTERMEDIATE CODE IMAGE DATA INTO RASTER IMAGE IN SEQUENCE OF EACH OF BANDS
318 PRINT OUT
END

[FIG. 13]

INTERMEDIATE CODE IMAGE DATA RECONSTRUCTION PROCESS 1
332 WHETHER OR NOT DOES UNPROCESSED EDGE LIST EXIST IN TRAGET SCANNING LINE?
334 SCAN EDGE LISTS
336 RECONSTRUCT INTERMEDIATE CODE IMAGE DATA
338 WHETHER OR NOT ARE ALL SCANNING LINES PROCESSED?
340 SET NEXT SCANNING LINE AS TARGET SCANNING LINE
RETURN

[FIG. 14]

INTERMEDIATE CODE IMAGE DATA RECONSTRUCTION PROCESS 2
352 SCAN FIRST EDGE LIST
354 RECONSTRUCT INTERMEDIATE CODE IMAGE DATA
356 WHETHER OR NOT IS THE NUMBER OF UNPROCESSED EDGE LISTS EQUAL TO OR GREATER THAN 2?
358 COMPARE DISTANCES TO TWO EDGE LISTS EACH OF WHICH MAY BECOME NEXT TRAGET EDGE LIST
360 SET SCANNING DIRECTION TO DIRECTION TOWARDS EDGE LIST HAVING SHORTER DISTANCE
362 SET UNPROCESSED EDGE LIST WHEN VIEWED IN SCANNING DIRECTION TO BE TARGET EDGE LIST
364 SCAN TARGET EDGE LIST
366 RECONSTRUCT INTERMEDIATE CODE IMAGE DATA
368 SCAN LAST EDGE LIST

370 RECONSTRUCT INTERMEDIATE CODE IMAGE DATA
RETURN

[FIG. 15]

INTERMEDIATE CODE IMAGE DATA RECONSTRUCTION PROCESS 3

382 WHETHER OR NOT DOES UNPROCESSED EDGE LIST EXIST IN TARGET SCANNING LINE?

384 SCAN EDGE LIST

386 RECONSTRUCT INTERMEDIATE CODE IMAGE DATA

388 WHETHER OR NOT IS TARGET SCANNING LINE LAST SCANNING LINE?

390 SET NEXT SCANNING LINE AS TARGET SCANNING LINE

392 SET FIRST SCANNING LINE AS TARGET SCANNING LINE

394 WHETHER OR NOT ARE ALL SCANNING LINES PROCESSED?

396 SET NEXT SCANNING LINE AS TARGET SCANNING LINE
RETURN

[FIGS. 16]

(A)

MAIN SCANNING LINE DIRECTION

SUB SCANNING LINE DIRECTION

EDGE LIST

MASK

(B)

MAIN SCANNING LINE DIRECTION

SUB SCANNING LINE DIRECTION

EDGE LIST

MASK

(C)

MAIN SCANNING LINE DIRECTION

SUB SCANNING LINE DIRECTION

EDGE LIST

MASK

[FIGS. 17]

(A)

MAIN SCANNING LINE DIRECTION
SUB SCANNING LINE DIRECTION
(B)
RUN LENGTH
DIFFERENCE IN Y DIRECTION
DIFFERENCE IN X DIRECTION

[FIGS. 18]
(A)
COUNT TABLE
POINTER TABLE
LINE 2
EFFECTIVE SCANNING LINE WIDTH
BAND WIDTH
EDGE LIST
MASK
(B)
COUNT TABLE
POINTER TABLE
EFFECTIVE SCANNING LINE WIDTH
BAND WIDTH
EDGE LIST
MASK

[FIG. 19]
MAIN SCANNING LINE DIRECTION
SUB SCANNING LINE DIRECTION
60 RENDERING OBJECT OF UPPER LAYER
50 RENDERING OBJECT OF LOWER LAYER

[FIG. 20]
MAIN SCANNING LINE DIRECTION
SUB SCANNING LINE DIRECTION
62 MASK (RENDERING OBJECT OF UPPER LAYER)
52 RENDERING OBJECT OF LOWER LAYER

[FIG. 21]

MAIN SCANNING LINE DIRECTION

SUB SCANNING LINE DIRECTION

64 MASK (RENDERING OBJECT AND IMAGE OF UPPER LAYER)

54 RENDERING OBJECT OF LOWER LAYER

[FIG. 22]

RETAINING BLOCK IN WORKING AREA

HEADER INFORMATION

OFFICIAL RETAINING BLOCK

HEADER INFORMATION

[FIG. 23]

START

402 SET BANDS

404 READ OUT EDGE LISTS OF OBJECT IN REVERSED SEQUENCE OF RENDERING SEQUENCE, AND DETECT RANGE OF Y COORDINATE (Y_S TO Y_E)

406 WHETHER OR NOT IS OBJECT PRESENT IN BANDS?

408 WHETHER OR NOT IS OBJECT IMAGE?

410 OVERLAPPED GRAPHICS PROCESSING

412 CREATE AND SAVE MASK EDGE LIST

414 TEMPORARILY RETAIN INTERMEDIATE CODE IMAGE DATA IN RETAINING BLOCK IN WORKING AREA

416 WHETHER OR NOT ARE ALL OBJECTS COMPLETED?

418 CHANGE RENDERING SEQUENCE ONLY IN CASE WHERE OBJECT IS IMAGE, AND RETAIN INTERMEDIATE CODE IMAGE DATA IN OFFICIAL RETAINING BLOCK

420 WHETHER OR NOT ARE ALL BANDS PROCESSED?

END

[FIGS. 24]

(A)

BAND INFORMATION

INTERMEDIATE CODE IMAGE DATA

RETAINING BLOCK

CONVENTIONAL METHOD
INTERMEDIATE CODE IMAGE DATA RETAINING BLOCK HEADER INFORMATION
96 INTERMEDIATE CODE IMAGE DATA OF IMAGE WHICH HAS BEEN SUBJECTED TO OVERLAPPED GRAPHICS PROCESSING
98 INTERMEDIATE CODE IMAGE DATA OTHER THAN IMAGE

(B)
BAND INFORMATION
INTERMEDIATE CODE IMAGE DATA
RETAINING BLOCK

PRESENT METHOD
INTERMEDIATE CODE IMAGE DATA RETAINING BLOCK HEADER INFORMATION
96 INTERMEDIATE CODE IMAGE DATA OF IMAGE WHICH HAS NOT BEEN SUBJECTED TO OVERLAPPED GRAPHICS PROCESSING
98 INTERMEDIATE CODE IMAGE DATA OTHER THAN IMAGE